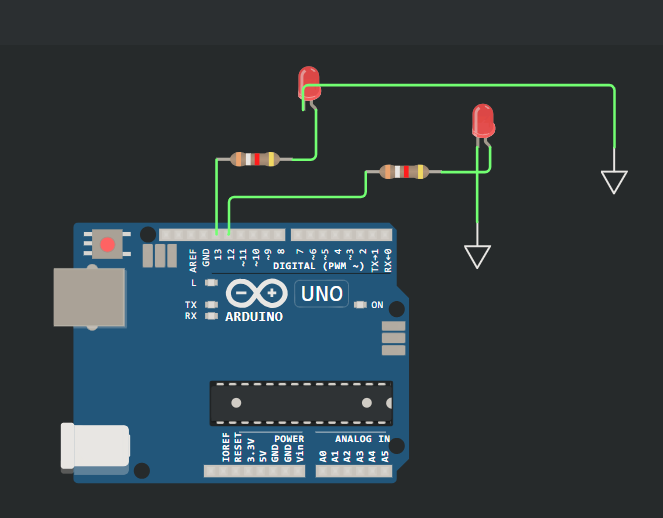
https://wokwi.com/dashboard/projects

Câu 1: a/ Lập trình điều khiển song song 2 LED: LED 1 bật 2s và tắt sau 3s; LED 2 bật trong 2.5s và tắt định kỳ sau mỗi 0.5s? Hiển thị trạng thái của các đèn LED?



#include <Arduino\_FreeRTOS.h>

// Định nghĩa chân của LED

#define LED1\_PIN 13

#define LED2\_PIN 12

// Prototype các hàm

void TaskLED1(void \*pvParameters);

void TaskLED2(void \*pvParameters);

void Taskprint(void \*pvParameters);

void setup() {

  // Khởi tạo serial communication với tốc độ 9600 bit/giây

**Serial**.begin(9600);

  // Khởi tạo các task

  xTaskCreate(TaskLED1, "LED1 Task", 128, NULL, 1, NULL); // Task điều khiển LED 1

  xTaskCreate(TaskLED2, "LED2 Task", 128, NULL, 1, NULL); // Task điều khiển LED 2

  xTaskCreate(Taskprint, "LED Status Task", 128, NULL, 1, NULL); // Task hiển thị trạng thái của các LED

  // Bắt đầu FreeRTOS

  vTaskStartScheduler();

}

void loop() {

  // Empty, all work is done in tasks

}

// Task điều khiển LED 1

void TaskLED1(void \*pvParameters) {

  pinMode(LED1\_PIN, OUTPUT); // Chân điều khiển LED 1

  while (1) {

    digitalWrite(LED1\_PIN, HIGH); // Bật LED 1

    vTaskDelay(2000 / portTICK\_PERIOD\_MS); // Chờ 2s

    digitalWrite(LED1\_PIN, LOW); // Tắt LED 1

    vTaskDelay(3000 / portTICK\_PERIOD\_MS); // Chờ 1s

  }

}

// Task điều khiển LED 2

void TaskLED2(void \*pvParameters) {

  pinMode(LED2\_PIN, OUTPUT); // Chân điều khiển LED 2

  while (1) {

    digitalWrite(LED2\_PIN, HIGH); // Bật LED 2

    vTaskDelay(2500 / portTICK\_PERIOD\_MS); // Chờ 2.5s

    digitalWrite(LED2\_PIN, LOW); // Tắt LED 2

    vTaskDelay(500 / portTICK\_PERIOD\_MS); // Chờ 0.5s

  }

}

// Task hiển thị trạng thái của các LED

void Taskprint(void \*pvParameters) {

  while (1) {

    // Đọc trạng thái của LED và hiển thị

    int LED1\_State = digitalRead(LED1\_PIN);

    int LED2\_State = digitalRead(LED2\_PIN);

**Serial**.print("LED 1: ");

**Serial**.print(LED1\_State);

**Serial**.print(", LED 2: ");

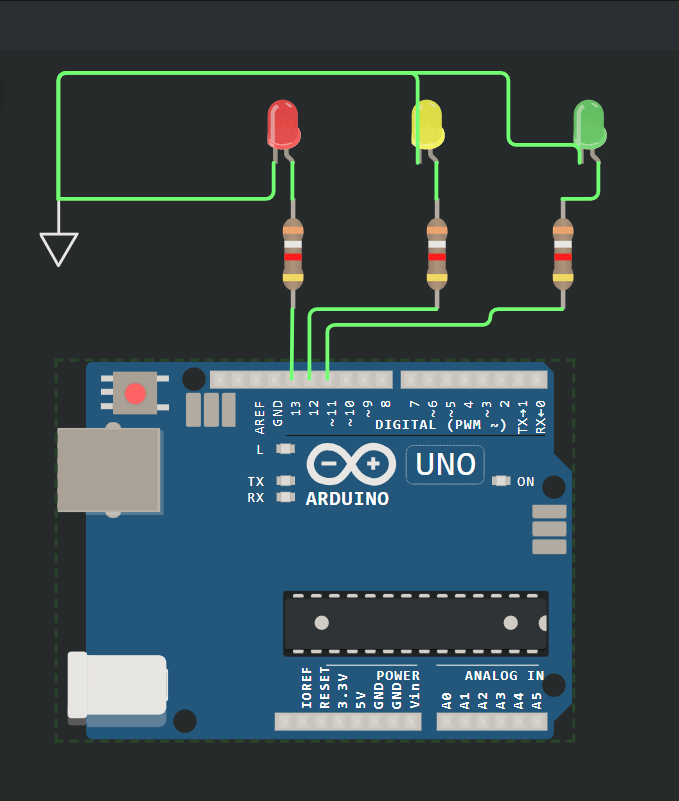
**Serial**.println(LED2\_State);

    vTaskDelay(1000 / portTICK\_PERIOD\_MS); // Cập nhật trạng thái mỗi giây

  }

}

Câu 1.b b. Lập trình điều khiển 3 LED mô phỏng tín hiệu đèn giao thông?



#include <Arduino\_FreeRTOS.h>

// Định nghĩa chân của các LED

#define RED\_LED\_PIN 13

#define YELLOW\_LED\_PIN 12

#define GREEN\_LED\_PIN 11

// Prototype các hàm

void TaskTrafficLights(void \*pvParameters);

void setup() {

  // Khởi tạo serial communication với tốc độ 9600 bit/giây

**Serial**.begin(9600);

  // Khởi tạo các task

  xTaskCreate(TaskTrafficLights, "Traffic Lights Task", 128, NULL, 1, NULL);

  // Bắt đầu FreeRTOS

  vTaskStartScheduler();

}

void loop() {

  // Empty, all work is done in tasks

}

// Task điều khiển tín hiệu đèn giao thông

void TaskTrafficLights(void \*pvParameters) {

  pinMode(RED\_LED\_PIN, OUTPUT);

  pinMode(YELLOW\_LED\_PIN, OUTPUT);

  pinMode(GREEN\_LED\_PIN, OUTPUT);

  while (1) {

    // Đèn xanh sáng trong 5 giây

    digitalWrite(RED\_LED\_PIN, LOW);

    digitalWrite(YELLOW\_LED\_PIN, LOW);

    digitalWrite(GREEN\_LED\_PIN, HIGH);

    vTaskDelay(10000 / portTICK\_PERIOD\_MS);

    // Đèn vàng sáng trong 2 giây

    digitalWrite(RED\_LED\_PIN, LOW);

    digitalWrite(YELLOW\_LED\_PIN, HIGH);

    digitalWrite(GREEN\_LED\_PIN, LOW);

    vTaskDelay(2000 / portTICK\_PERIOD\_MS);

     // Đèn đỏ sáng trong 5 giây

    digitalWrite(RED\_LED\_PIN, HIGH);

    digitalWrite(YELLOW\_LED\_PIN, LOW);

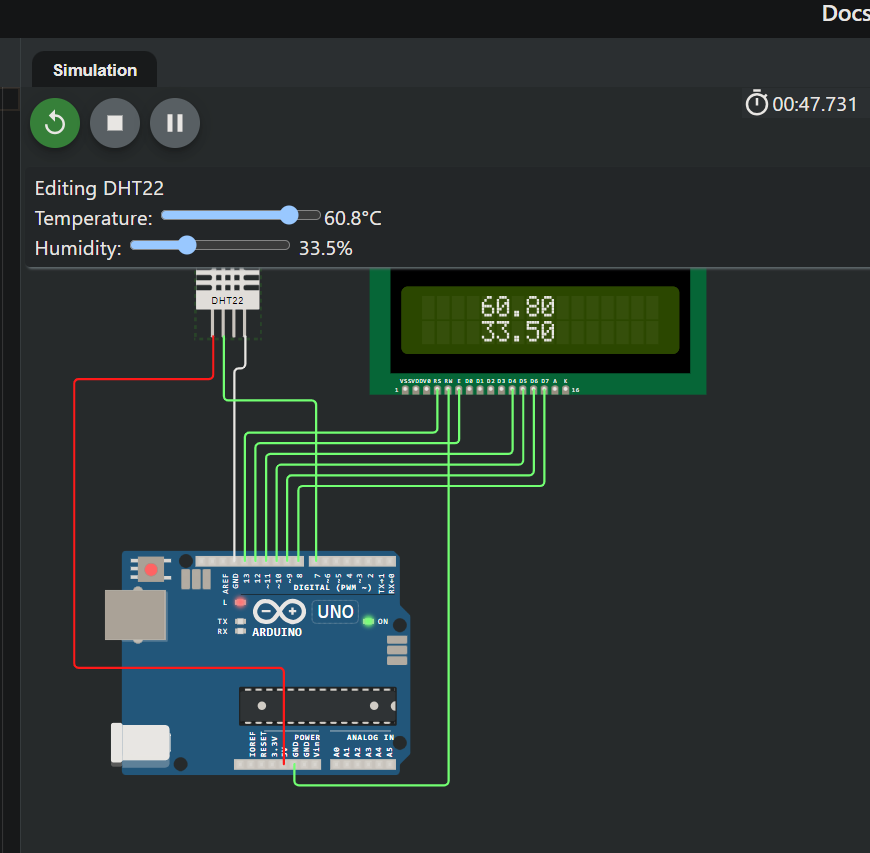
    digitalWrite(GREEN\_LED\_PIN, LOW);

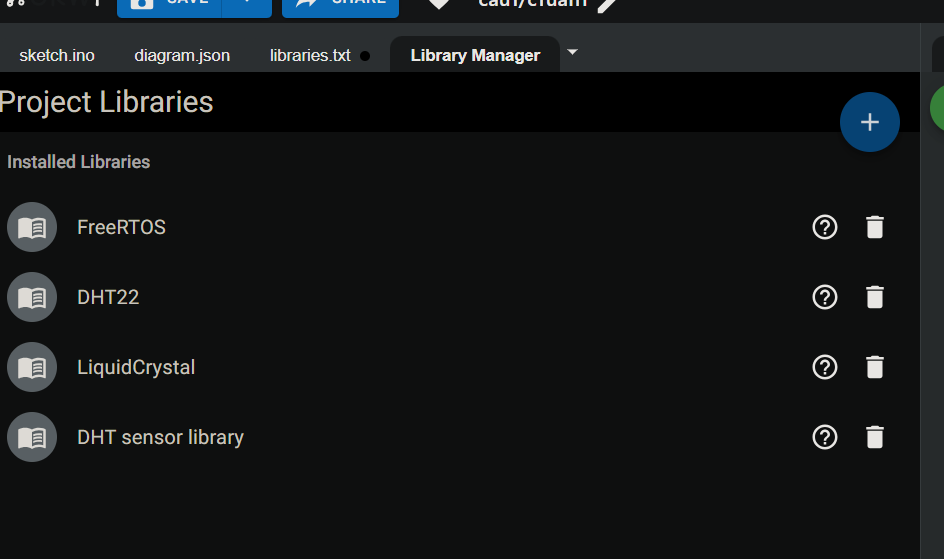
    vTaskDelay(5000 / portTICK\_PERIOD\_MS);

  }

}

Câu 1/ C





#include<LiquidCrystal.h>

LiquidCrystal lcd(13,12,11,10,9,8);

#include <DHT.h>

#define DHTPIN 7

#define DHTTYPE DHT22

DHT dht(DHTPIN,DHTTYPE);

void setup(){

  lcd.begin(16,2);

  lcd.setCursor(0,0);

  dht.begin();

  lcd.print("TEMPERATURE");

  lcd.setCursor(2,1);

  lcd.print("SENSOR DHT22");

  delay(1000);

}

void loop(){

  float h= dht.readHumidity();

  float t= dht.readTemperature();

  lcd.clear();

  lcd.setCursor(4,0);

  lcd.print(t);

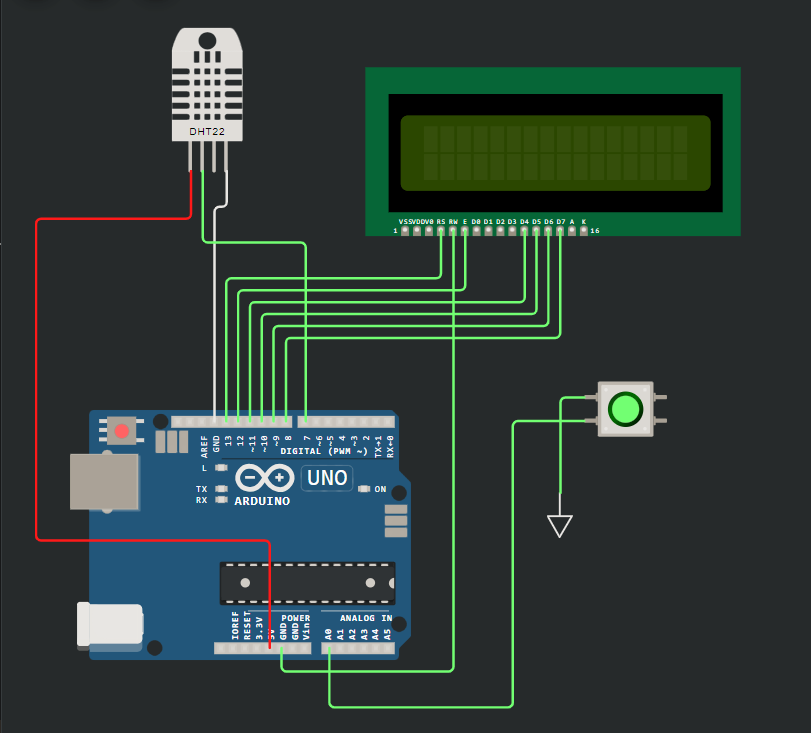
  lcd.setCursor(4,1);

  lcd.print(h);

  delay(250);

}

Câu 1/d



#include<LiquidCrystal.h>

LiquidCrystal lcd(13,12,11,10,9,8);

#include <DHT.h>

#define DHTPIN 7

#define DHTTYPE DHT22

DHT dht(DHTPIN,DHTTYPE);

#define BUTTON\_PIN A0

float maxTemperature = -100; // Khởi tạo nhiệt độ lớn nhất

float maxHumidity = -100;    // Khởi tạo độ ẩm lớn nhất

void setup(){

  pinMode(A0, INPUT\_PULLUP);

  lcd.begin(16,2);

  lcd.setCursor(0,0);

  dht.begin();

  lcd.print("TEMPERATURE");

  lcd.setCursor(2,1);

  lcd.print("SENSOR DHT22");

  delay(1000);

}

void loop(){

  float h= dht.readHumidity();

  float t= dht.readTemperature();

  lcd.clear();

  lcd.setCursor(4,0);

  lcd.print(t);

  lcd.setCursor(4,1);

  lcd.print(h);

  delay(250);

  if (digitalRead(A0) == LOW) {

    // Tìm nhiệt độ và độ ẩm lớn nhất

    maxTemperature = max(maxTemperature, t);

    maxHumidity = max(maxHumidity, h);

    // Hiển thị nhiệt độ và độ ẩm lớn nhất trên màn hình LCD

    lcd.clear();

    lcd.setCursor(0, 0);

    lcd.print("Max Temp: ");

    lcd.print(maxTemperature);

    lcd.print(" C");

    lcd.setCursor(0, 1);

    lcd.print("Max Humidity: ");

    lcd.print(maxHumidity);

    lcd.print(" %");

    // Chờ một lúc trước khi trở về hiển thị dữ liệu thông thường

    delay(2000);

  } else {

    // Nếu nút không được nhấn, tiếp tục hiển thị dữ liệu như bình thường

    delay(250);

  }

}

Câu 2

Tạo file VLUIoT.h

#ifndef VLUIoT\_h

#define VLUIoT\_h

#include "Arduino.h"

class VLUIoT {

  public:

    VLUIoT(int pin);

    void on(int duration);

    void off(int duration);

  private:

    int \_pin;

};

#endif

Tạo file VLUIot.cpp

#include "Arduino.h"

#include "VLUIoT.h"

VLUIoT::VLUIoT(int pin) {

  pinMode(pin, OUTPUT);

  \_pin = pin;

}

void VLUIoT::on(int duration) {

  digitalWrite(\_pin, HIGH);

  delay(duration);

}

void VLUIoT::off(int duration) {

  digitalWrite(\_pin, LOW);

  delay(duration);

}

Code

#include "VLUIoT.h"

VLUAVG sensorData;

VLUIoT led(13);

void setup() {

  // Không cần thiết lập gì thêm

**Serial**.begin(9600);

  dht.begin();

}

void loop() {

  led.on(4000);  // Bật LED trong 1 giây

  led.off(1000); // Tắt LED trong 1 giây

}

2b

VLUAVG.h

#ifndef VLUAVG\_h

#define VLUAVG\_h

#include "Arduino.h"

class VLUAVG {

  public:

    VLUAVG(); // Constructor

    void addTemperature(float temperature); // Thêm giá trị nhiệt độ

    void addHumidity(float humidity);       // Thêm giá trị độ ẩm

    float getAverageTemperature();           // Trả về giá trị trung bình của nhiệt độ

    float getAverageHumidity();              // Trả về giá trị trung bình của độ ẩm

    int getCount();                          // Trả về số lượng giá trị đã lưu trữ

    void clear();                            // Xóa dữ liệu đã lưu trữ

  private:

    float \_temperatures[10]; // Mảng lưu trữ các giá trị nhiệt độ

    float \_humidities[10];   // Mảng lưu trữ các giá trị độ ẩm

    int \_count;              // Số lượng giá trị đã lưu trữ

};

#endif

VLUAVG.cpp

#include "VLUAVG.h"

VLUAVG::VLUAVG() {

  \_count = 0;

}

void VLUAVG::addTemperature(float temperature) {

  if (\_count < 10) {

    \_temperatures[\_count] = temperature;

    \_count++;

  }

}

void VLUAVG::addHumidity(float humidity) {

  if (\_count < 10) {

    \_humidities[\_count] = humidity;

    \_count++;

  }

}

float VLUAVG::getAverageTemperature() {

  float sum = 0;

  for (int i = 0; i < \_count; i++) {

    sum += \_temperatures[i];

  }

  return sum / \_count;

}

float VLUAVG::getAverageHumidity() {

  float sum = 0;

  for (int i = 0; i < \_count; i++) {

    sum += \_humidities[i];

  }

  return sum / \_count;

}

int VLUAVG::getCount() {

  return \_count;

}

void VLUAVG::clear() {

  \_count = 0;

}

Code

#include "VLUIoT.h"

#include "VLUAVG.h"

#include "DHT.h"

// Định nghĩa chân kết nối cảm biến DHT

#define DHTPIN 4

#define DHTTYPE DHT22

// Khởi tạo các đối tượng

DHT dht(DHTPIN, DHTTYPE);

VLUAVG sensorData;

VLUIoT led(13);

void setup() {

  // Khởi tạo Serial Monitor

**Serial**.begin(9600);

  // Khởi tạo cảm biến DHT

  dht.begin();

}

void loop() {

  // Đọc nhiệt độ và độ ẩm từ cảm biến DHT22

  float temperature = dht.readTemperature();

  float humidity = dht.readHumidity();

  // In giá trị nhiệt độ và độ ẩm lên Serial Monitor

**Serial**.print("Temperature: ");

**Serial**.println(temperature);

**Serial**.print("Humidity: ");

**Serial**.println(humidity);

  // Lưu trữ giá trị nhiệt độ và độ ẩm vào thư viện VLUAVG

  sensorData.addTemperature(temperature);

  sensorData.addHumidity(humidity);

  // Kiểm tra nếu đã đủ 10 giá trị, sau đó tính và in ra trung bình nhiệt độ và độ ẩm

  if (sensorData.getCount() >= 10) {

**Serial**.print("Average Temperature: ");

**Serial**.println(sensorData.getAverageTemperature());

**Serial**.print("Average Humidity: ");

**Serial**.println(sensorData.getAverageHumidity());

    // Xóa dữ liệu để tính lại trung bình cho 10 giá trị tiếp theo

    sensorData.clear();

  }

  delay(2000); // Chờ 2 giây trước khi đọc dữ liệu tiếp theo

}